Network Situational Awareness with tcpdump

Thomas Phillips

Why tcpdump?

Because it’s everywhere!
/usr/sbin/tcpdump
(and WinDump, too)

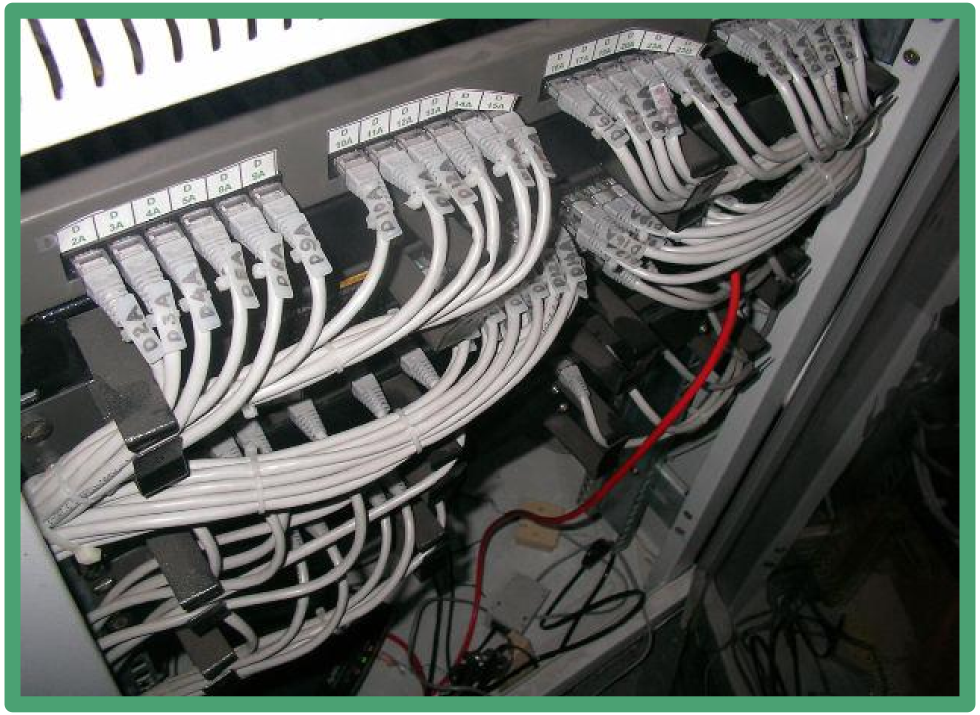
http://cdn.morguefile.com/imageData/public/files/e/Elfaramawy/10/l/1414048297276uo.jpg

Topics

The Network Stack

tcpdump Fundamentals

Interesting Recipes



The Network Stack

The Essential Protocols

Ethernet

ARP

Internet Protocol

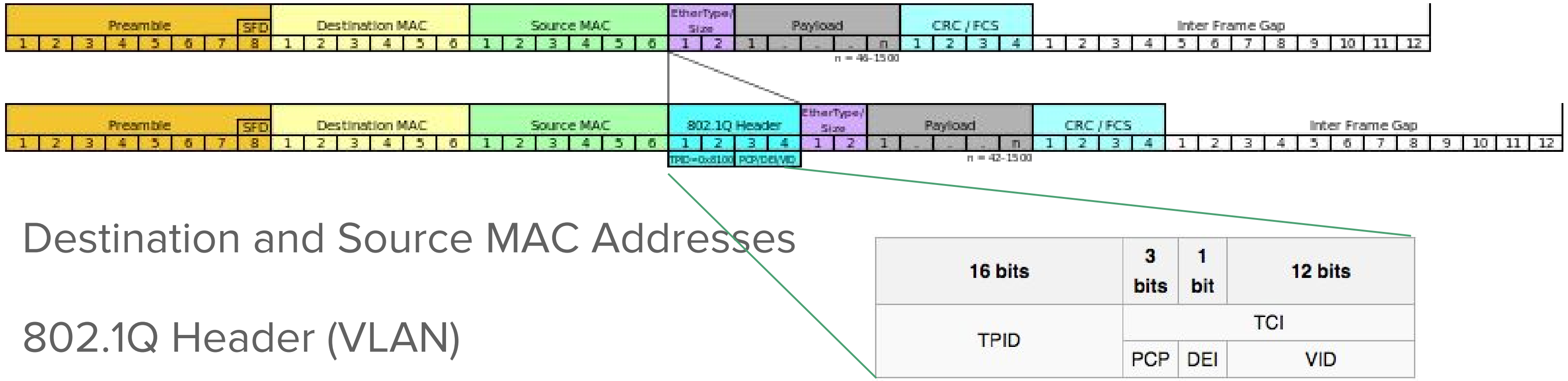
ICMP

TCP

UDP

Ethernet

https://en.wikipedia.org/wiki/IEEE_802.1Q



16 bits	3 bits	1 bit	12 bits
TPID	TCI		
	PCP	DEI	VID

Destination and Source MAC Addresses

802.1Q Header (VLAN)

EtherType (https://www.iana.org/assignments/ieee-802-numbers/ieee-802-numbers.xhtml)

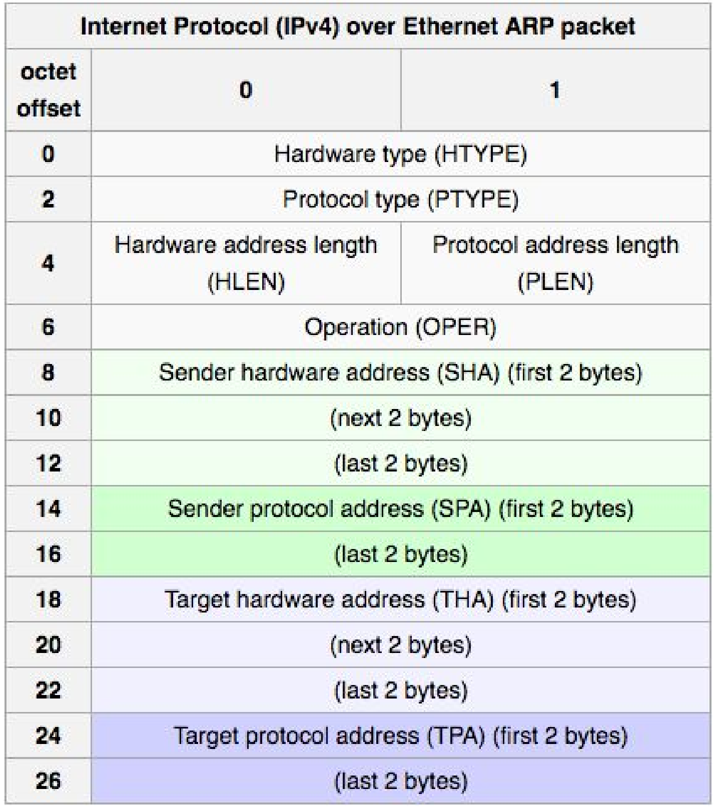
Payload

Address Resolution Protocol (ARP)

Hardware and Protocol Types

Sender and Target Hardware Addresses

Sender and Target Protocol Addresses

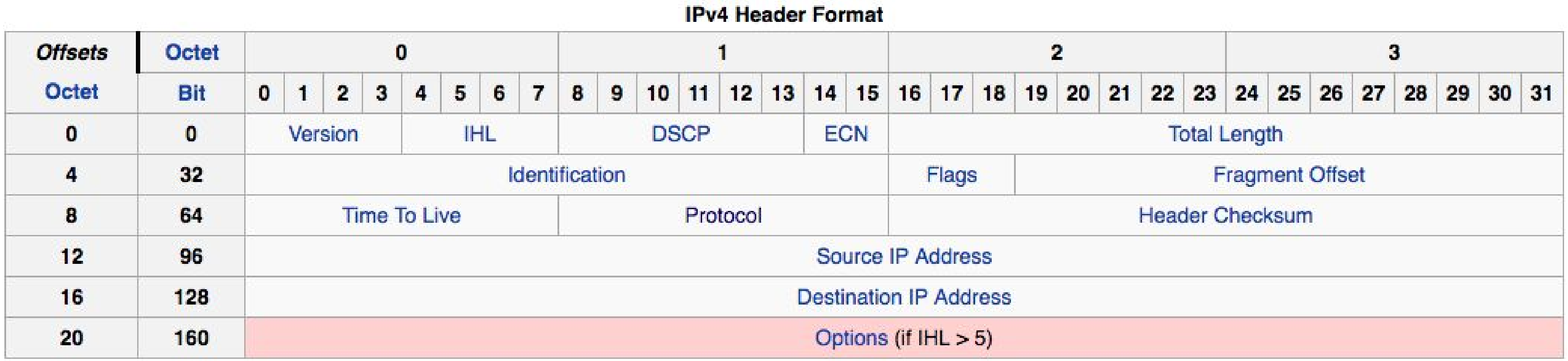


Internet Protocol (IPv4) over Ethernet ARP packet

octet offset	0	1
0	Hardware type (HTYPE)	
2	Protocol type (PTYPE)	
4	Hardware address length (HLEN)	Protocol address length (PLEN)
6	Operation (OPER)	
8	Sender hardware address (SHA) (first 2 bytes)	
10	(next 2 bytes)	
12	(last 2 bytes)	
14	Sender protocol address (SPA) (first 2 bytes)	
16	(last 2 bytes)	
18	Target hardware address (THA) (first 2 bytes)	
20	(next 2 bytes)	
22	(last 2 bytes)	
24	Target protocol address (TPA) (first 2 bytes)	
26	(last 2 bytes)	

https://en.wikipedia.org/wiki/Address_Resolution_Protocol

Internet Protocol (IPv4)


IPv4 Header Format

Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Version				IHL				DSCP						ECN		Total Length															
4	32	Identification																Flags			Fragment Offset												
8	64	Time To Live								Protocol								Header Checksum															
12	96	Source IP Address																															
16	128	Destination IP Address																															
20	160	Options (if IHL > 5)																															

IHL (x 4)

Total Length (including header)

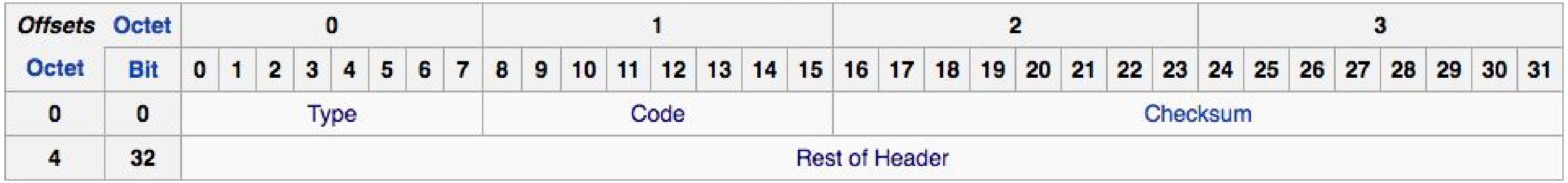
Time To Live (TTL)

Protocol

Source and Destination Addresses

Internet Control Message Protocol (ICMP)

ICMP Header Format



Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Type								Code								Checksum															
4	32	Rest of Header																															

https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol

Type 8, Code 0 = Echo Request

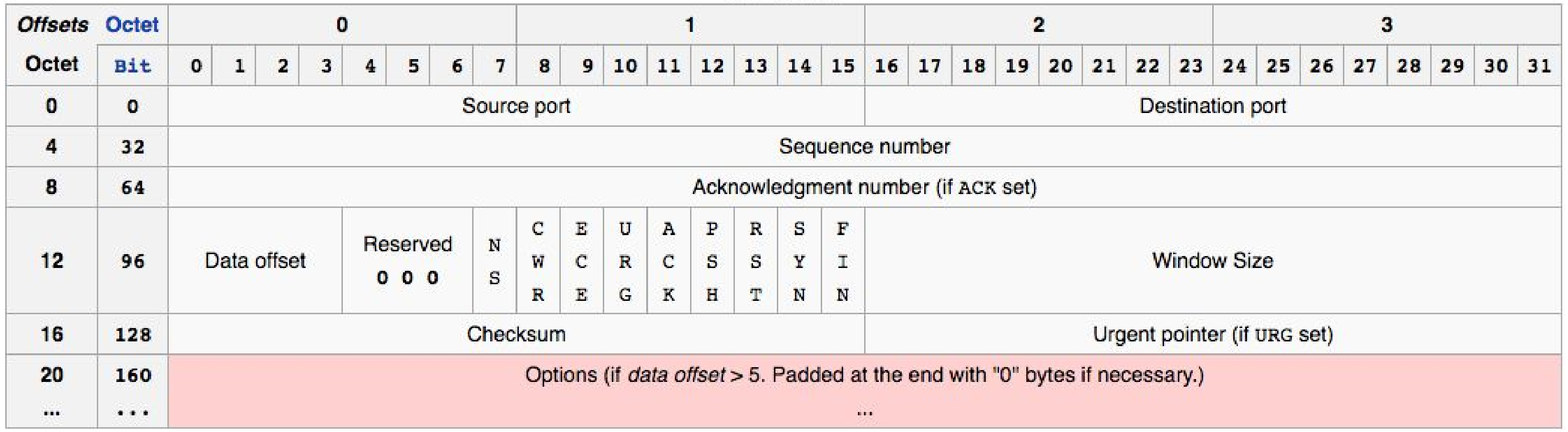
Type 0, Code 0 = Echo Reply

Type 3 = Destination Unreachable

Type 11 = Time Exceeded

Transmission Control Protocol (TCP)

TCP Header



Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Sequence number																															
8	64	Acknowledgment number (if ACK set)																															
12	96	Data offset				Reserved 0 0 0			NS	CWR	ECE	URG	ACK	PSH	RST	SYN	FIN	Window Size															
16	128	Checksum																Urgent pointer (if URG set)															
20 ...	160 ...	Options (if *data offset* > 5. Padded at the end with "0" bytes if necessary.) ...																															

https://en.wikipedia.org/wiki/Transmission_Control_Protocol

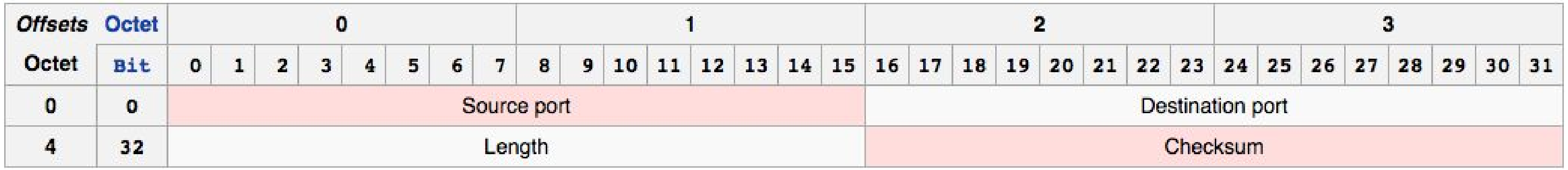
Source and Destination Ports

Data Offset (x 4)

Flags (SYN, FIN, RST, PSH)

User Datagram Protocol (UDP)

UDP Header



Offsets	Octet	0								1								2								3							
Octet	Bit	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
0	0	Source port																Destination port															
4	32	Length																Checksum															

https://en.wikipedia.org/wiki/User_Datagram_Protocol

Source and Destination Ports

Length (including header)

tcpdump Fundamentals

The Essential Options

http://cdn.morguefile.com/imageData/public/files/r/ronnieb/preview/fldr_2008_03_13/file0002113164146.jpg

The Basics

Be promiscuous: ifconfig eth0 promisc

Run as root: sudo tcpdump

Useful options: -F, -i, -n, -nn, -q, -r, -s, -v, -vv, -vvv, -w

Pick a protocol: ether, arp, ip, icmp, tcp, udp (there are others, too)

Filter for addresses, if applicable.

Filter for ports, if applicable.

Filter for specific fields.

Operations:

    (,),
    and, or, not,
    =, !=, <, <=, >, >=,
    &, |, <<, >>, +, -

Useful options

-F *file* Read the filter expression from the file.

-i *iface* Listen on a specific interface.

-n Do not do DNS lookups.

-nn Do not use port labels.

-q Quiet mode.

-r *file* Read from a pcap file.

-s *snaplen* Snarf snaplen bytes per packet.

-v, -vv, -vvv Levels of verbosity.

-w *file* Write to a pcap file.

Core Filters

Example: `sudo tcpdump arp src host 192.168.1.1`

Protocol:

- ether
- arp
- ip
- icmp
- tcp
- udp

Address:

- host *address*
- src host *address*
- dst host *address*

Port:

- port *num*
- src port *num*
- dst port *num*

Filter for Specific Fields

protocol[start:length] op value

IPv4 Time To Live

```
ip[8]
```

IPv4 Total Length

```
ip[2:2]
```

IPv4 Header Length

```
(ip[0] & 0xf) << 2
```

TCP Data Offset

```
(tcp[12] & 0xf0) >> 2
```

TTL is less than 3

TTL ip[8] < 3

TCP segments that contain data

Total Length Header Length

```
(
 (
  (ip[2:2] - (ip[0]&0xf)<<2) -
  ((tcp[12]&0xf0)>>2)
 )
 != 0
)
```

Data Offset

Interesting Recipes

Good Stuff

http://cdn.morguefile.com/imageData/public/files/m/Moonlightway/10/l/1414561609wqz2o.jpg

Interesting Recipes

Traffic not on my VLAN (data leakage)

Traceroute activity

Fragmented IP traffic

TCP traffic leaving a subnet

Unauthorized TCP ports

List of endpoints calling this endpoint (TCP, UDP)

List of endpoints this endpoint is calling out to (TCP, UDP)

HTTP POST requests

Traffic not on my VLAN (data leakage)

```
tcpdump \
    -vv \
    -i eth0 \
    '( vlan and ( ether[14:2] & 0xfff != 1000 ) )'
```

Look for 802.1Q frames tagged with VLAN

Adapted from:
http://serverfault.com/questions/196250/tcpdump-capture-one-of-several-vlans

Traceroute activity

```
tcpdump \
    -vv \
    -i eth0 \
    '( ip[8] < 3 )'
```

Low TTL

Protocol other than IPv4 is not important.

Fragmented IP traffic

```
tcpdump \
    -vv \
    -i eth0 \
    '( (ip[6] & 64 == 0) and (ip[6:2] > 0))'
```

Don’t fragment bit is not set.

Fragment Offset

Adapted from:
https://blog.wains.be/2007/2007-10-01-tcpdump-advanced-filters.md

TCP traffic leaving a subnet

```
tcpdump \
    -vv \
    -i eth0 \
    '(tcp[13] & 18 == 2) and not dst net 192.168.1.0/24'
```

SYN flag set, no ACK

Our subnet

Unauthorized TCP ports

```
tcpdump \

    -vv \

    -i eth0 \

    'dst host 192.168.1.100 and \
     (tcp[13] & 18 == 2) and (not dst port 22)'
```

Our IP address

SYN flag set, no ACK

Display anything not SSH.

List of endpoints calling this endpoint (TCP, UDP)

-l for line buffered

```
tcpdump \
    -i eth0 -l \
    '(udp or (tcp[13] & 18 == 2)) and \
    dst host 192.168.1.100 | cut -d' ' -f3,6-
```

Example output:

```
zambonia.domain 18870* 1/0/0 PTR Zambonia. (64)
qb-in-f189.1e100.net.https UDP, length 41
qb-in-f189.1e100.net.https UDP, length 20
qh-in-f189.1e100.net.https UDP, length 36
qb-in-f189.1e100.net.https UDP, length 33
qb-in-f189.1e100.net.https UDP, length 63
qb-in-f189.1e100.net.https UDP, length 39
zambonia.domain 65529 2/0/0 CNAME fd-fp3.wg1.b.yahoo.com., AAAA 2001:4998:58:c02::a9 (86)
```

List of endpoints this endpoint is calling out to (TCP, UDP)

```
tcpdump \
    -i eth0 -l -nn \
    '(udp or (tcp[13] & 18 == 2)) and \
     not dst net 192.168.1.0/24' | cut -d' ' -f5-
```

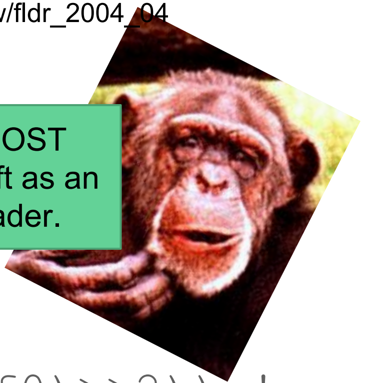
-l for line buffered

Example output:

```
74.125.22.189.443: UDP, length 38
98.139.180.149.80: Flags [S], seq 1112211606, win 65535, options [mss 1460,nop,wscale 5,
nop,nop,TS val 440461606 ecr 0,sackOK,eol], length 0
172.217.1.195.443: UDP, length 1350
172.217.1.195.443: UDP, length 337
172.217.1.195.443: Flags [S], seq 640180255, win 65535, options [mss 1460,nop,wscale 5,nop,
nop,TS val 440462066 ecr 0,sackOK,eol], length 0
172.217.1.195.443: UDP, length 37
172.217.1.195.443: UDP, length 40
```

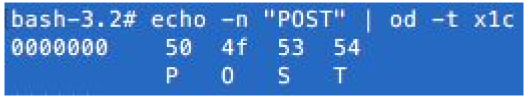
HTTP POST requests

http://cdn.morguefile.com/imageData/public/files/s/seeka/preview/fldr_2004_04_14/file0001535568481.jpg



Searching for the POST string in the filter is left as an exercise for the reader.

```
tcpdump \
    -i eth0 -l \
    'tcp port 80 and \
     (((ip[2:2] - ((ip[0]&0xf)<<2)) - ((tcp[12]&0xf0)>>2)) !=
0)' | egrep --line-buffered 'POST'
```



```
bash-3.2# echo -n "POST" | od -t x1c
0000000    50   4f   53   54
           P    O    S    T
```

Adapted from:
https://blog.wains.be/2007/2007-10-01-tcpdump-advanced-filters.md

Example output:

```
23:51:13.012659 IP dunwich.hsd1.md.comcast.net.52660 > us-3-zone-1.syronex.com.http: Flags
[P.], seq 1451860020:1451860913, ack 4079266449, win 4117, options [nop,nop,TS val
442006656 ecr 356004271], length 893: HTTP: POST /form/yx/submit HTTP/1.1
```

El fin.